주 제

통신망에서 폴리모픽 웜 공격의 탐지 기술

대구가톨릭대학교 전용희, 장정숙, 한국전자통신연구원 장종수, 남택용

차 례

I. 서 론

인터넷 웜은 목표 시스템 상에서 수행되는 서비스의 취약성을 이용하여 많은 수의 호스트에 영향을 미치기 위해서 네트워크 사이를 자동적으로 확산해 나가는 악성 프로그램이다. 웜의 탐지방법에서 시그너처(signature) 탐지는 잠재적인 악성 트래픽을 식별하기 위하여 네트워크 트래픽에 대한 알려진 시그너처를 이용한다. 이를 위하여 다수의 웜-관련 네트워크 트래픽 스트림을 분석하여 인터넷 웜을 탐지하기 위한 시그너처를 자동으로 생성한다. 대부분의 침입탐지시스템(IDS: Intrusion Detection System)들은 신속한 침입탐지와 분석의 용이, 오탐지(false positive)의 최소화를 위하여 시그너처 탐지방법을 이용한다.

시그너처 탐지 방법에서 시그너처 매칭 기법은 네트워크 트래픽 안의 특별한 비트 패턴 인 스트링 매칭에 의존한다. 이 방법은 웜 코드가 감염 과정동안 변하지 않는다는 가정을 기반으로 하고 있다. 그러나 미래 인터넷 웜의 새로운 경향으로 폴리모픽(Polymorphic) 웜이 제시되고 있다. 즉 웜은 네트워크를 따라 확산되면서 변화(mutate) 할 수 있기 때문에, 탐지를 피할 수 있다.

2001년 초반에 공격자로 하여금 어떤 종류의 서비스에 대한 버퍼 오버 플로 공격을 숨기도록 하는 새로운 도구가 공개되었다. 이 도구는 K2라는 익명의 저자가 작성한 ADMutate폴리모픽 엔진으로, 실행될 때마다 익스플로잇 시그너처를 변경하는 "polymorphic shellcode"를 생성한다[1]. 셸 코드는 악성 코드를 주입하기 위하여 버퍼 오버 플로를 이용하며, 보통 NOP 존(zone), shellcode, 반환 주소(return address) 존으로 구성된다. 일반적인 셸 코드는 이

러한 특성으로 인하여 단순한 시그너처에 의하여 탐지될 수 있다. 따라서 셀 코드를 암호화 한다든지, 같은 결과를 가져다주는 다른 함수들을 사용하여 변환시킨다. 이와 같이 어떤 프로그램을 같은 기능을 가진 다른 코드로 구성된 하나의 버전으로 변환하기 위하여 사용하는 컴퓨터 프로그램을 폴리모픽(혹은 mutation) 엔진이라 한다.

본 논문에서는 비정상 트래픽을 발생시키는 새로운 네트워크 공격 유형으로 폴리모픽 웜에 대하여 기술하고자 한다. 먼저 II장에서는 폴리모픽 웜의 정의와 구조, 폴리모픽 웜의 구현 방법에 대하여 기술하고, III장에서는 폴리모픽 엔진에 대해서 분석하고, IV장에서는 폴리모픽 웜을 탐지하는 기술로서 지문생성(Finger-Printing) 기술과 의미론-인지(Semantics-Aware) 기술에 대하여 알아본다. 마지막으로 V장에서는 결론 및 향후 연구로서 끝을 맺는다.

II. 폴리모픽 웜

2.1 정의와 구조

웜의 통상적인 정의는 "네트워크 사이를 확산하는 자기-복제 코드의 조각이며 전파를 위해 인간의 중재를 필요로 하지 않는다."는 것이다. 웜의 기본 컴포넌트로는 웜탄두(warhead), 전파엔진(Propagation Engine), 목표 선택 알고리즘(Target Selection Algorithm), 스캐닝 엔진(Scanning Engine) 그리고 페이로드(Payload)로 구성되며, 아래와 같이 동작한다.

- 웜 탄두 : 목표 시스템 상의 버퍼 오버플로 익스플로잇, 파일 공유 공격, E-mail, 통상적인 잘못된 구성(misconfiguration)과 같은 취약점을 이용하는 코드의 조각으로 익스플로잇(exploit)이라 한다.
- 전파엔진 : FTP, TFTP, HTTP, SMB (Server Message Block)와 같은 전파 메커니즘을 사용하여 희생 머신 상에 명령을 수행할 수 있도록 웜 코드를 희생시스템으로 전파 시킨다.
- 목표 선택 알고리즘 : 희생머신 상에서 새로운 희생머신을 검색하고 취약 머신을 결정하기 위해 목표 선택 알고리즘이 주소를 스캔 한다. 전자메일주소, 호스트 목록, 신뢰시스템, 네트워크 이웃, DNS 질의 등을 이용하여 목표를 선택한다.
- 스캐닝 엔진 : 스캐닝 엔진을 이용하여 웜은 해당 머신 상에서 웜 익스플로잇 동작 여부를 측정하기 위하여 웜 패킷들을 보내게 된다. 적절한 타깃을 발견한 후, 희생자에게 웜을 확산하고 전체 전파과정을 반복한다.
- 페이로드 : 목표 시스템 상에 특정 행동을 구현하기위해 설계된 코드이며, 어떤 웜들은 다른 머신으로 확산만 할 뿐 목표 시스템에 대하여 아무 일도 수행하지 않는다. 이 경우에는 페이로드가 없다.

페이로드에 포함되어 수행될 수 있는 선택 사항들로 백도어 개설, 분산 DoS 플러드 에이전트 심기, 복잡한 수학 연산 수행, 머신 재구성, 파일 제거, 웹 사이트 훼손 등과 같은 공격이 있다. 이외에도 웜 페이로드는 공격자가 원하는 타깃 시스템 상에 어떤 일이라도 할 수 있다. 페이로드의 공격 속성을 정리하면 다음과 같은 14개의 일반적인 공격 속성으로 분류 될 수 있다(<표 1> 참조) [2].

폴리모픽 웜(PW: Polymorphic Worm)은 침입탐지를 회피하고, 역공학 분석을 따돌리고, 필터를 통과하기 위하여 외양을 변경하는 웜을 의미한다. 다른

〈표 1〉 페이로드의 공격 속성

공격 속성			
취약 네트워크 코드 이용	사용자 속임	취약 구성 이용	사전에 설치된 백도어 이용
파일 시스템 변경	시스템 설정 변경	프로세스 수정	네트워크 접근
향상된 특권 요구	비정상 질의 수행	중요 API 호출	네트워크 범람 야기
지역 시스템 지연	웜 시그너처 포함		

웜 인스턴스의 바이트 시퀀스를 완전하게 다르게 보이게 하는 웜이다. 폴리모픽 프로그램은 소프트웨어 코드를 스크램블링하여 외양을 동적으로 변경할 수 있으며, 새로운 소프트웨어가 완전히 다른 명령어로 구성되어 있을지라도 그 코드는 여전히 정확히 같은 기능을 가진다. 폴리모피즘은 단지 외양만 변경되고 코드의 기능은 변경되지 않으며, 웜의 페이로드가 탐지 시그너처와 일치하지 않도록 전체 웜을 다른 변화된 버전으로 자동적으로 변형시킨다. 웜이 폴리모픽으로 되면 각 세그먼트는 on the fly로 새로운 코드가 생성되며, 다른 외양을 가져 탐지를 어렵게 만든다. 그러므로 같은 기능을 가지면서 수백만 개의 유일한 웜 세그먼트가 네트워크에 산재 할 것이다.

통상적인 방법으로는 원래의 웜 코드를 난수 키를 가지고 암호화하고, 키를 위해 매번 인스턴스와 함께 변경되는 복호기를 생성한다. 폴리모픽 엔진에 의해 수행되는 웜 코드는 변경되지 않으며, 정교한 폴리모픽 웜은 자신뿐만 아니라 익스플로잇도 변경할 수 있다. 이런 폴리모픽 웜의 가능한 구성요소는 다음과 같다[3].

- 시스템 침투용 공격 벡터 : 정교한 웜은 스택(stack), 힙(heap) 오버 플로 형태 익스플로잇, 백도어, 패스워드 스니핑, 중간자 공격 등과 같은 공격 백터를 이용한다.
- 공격 벡터 불변요소 : 폴리모픽 엔진은 공격의 어떤 부분이 취약한지를 결정하기 위해 공격 불변요소를 사용하여 공격을 막지 않고 변경될 수 있도록 한다.
- 폴리모픽 엔진 : 폴리모픽 복호기와 공격의 변환된 버전을 생성한다.
- 웜 몸체 코드 : 공격 벡터를 선정하고 목적지의 집합을 생성하며, 폴리모픽 엔진을 이용하여 공격 및 자신을 변환시키는 그리고 나서 변환된 인스턴스를 전송하는 코드를 가지고 있다.

2.2 폴리모픽 웜의 구현

가장 먼저 알려진 폴리모픽 바이러스는 Mark

```
Start:
GOTO Decryption_Code
Encrypted:
  ...
  lots of encrypted code
  ...
Decryption_Code:
  *A= Encrypted
Loop:
  B= *A
  B= B XOR CryptoKey
  *A= B
  A = A + 1
  GOTO Loop IF NOT A = (Decryption_Code-Encrypted)
  GOTO Encrypted
CryptoKey:
  some_random_number
```

(그림 1(a)) 원래의 알고리즘

```
Start:
GOTO Decryption_Code
Encrypted:
  ...
  lots of encrypted code
  ...
Decryption_Code:
  C = C + 1
  *A= Encrypted
Loop:
  B= *A
  C = 3214 * A
  B= B XOR CryptoKey
  *A= B
  C = 1
  C = A + B
  A = A + 1
  GOTO Loop IF NOT A = (Decryption_Code-Encrypted)
  C = C^2
  GOTO Encrypted
CryptoKey:
  some_random_number
```

(그림 1(b)) 불필요한 C-변경코드를 가진 동일 알고리즘

Washburn에 의해 1990년에 작성되었으며, 그 후 1992년에 Dark Avenger에 의하여 더 많이 알려진 폴리모픽 바이러스가 만들어 졌다. (그림 1)은 아주 단순한 폴리모픽 코드의 예를 보여준다. (그림 1(a)) 은 원래의 알고리즘이고, (그림 1(b))은 많은 불필요한 C-변경 코드를 가진 동일 알고리즘이다. 변수 A와 B를 사용하고 변수 C를 사용하지 않는 알고리즘이 변수 C의 내용을 변화시킨 많은 코드를 추가해도 알고리즘이 그대로 남아있음을 보여준다.

폴리모픽 웜의 초보적인 형태로 먼저 Klez 웜이 있으며, 이는 마이크로소프트의 아웃룩 전자메일을 통하여 전파되고 전자메일 스팸 필터를 회피하기 위해 전자메일 제목 라인을 변경하였다. Nimda 전자메일 분배 벡터도 또한 제목라인을 변경하였다. ADMutate란 도구는 웜 안에 있는 모든 코드를 변환하기 위한 변환 엔진으로, 버퍼 오버 플로 익스플로잇을 변환하는데 사용된다. Hydan 도구는 융통성 있는 폴리모픽 코드를 구현한다. 앞으로의 웜은 AD Mutate와 Hydan 등과 같은 폴리모픽 엔진을 이용하여 완전한 폴리모픽 웜의 형태로 생성될 것으로 판단된다.

폴리모픽 엔진은 mutation engine 혹은 mutating engine 이라 부르며, 어떤 프로그램을 같은 기능을 가지는 다른 코드로 이루어진 버전으로 변환하기 위하여 사용될 수 있는 컴퓨터 프로그램을 의미한다. 폴리모픽 엔진은 자신이 감염시키는 모든 프로그램에 대해서 다른 방법으로 스스로를 암호화하여, 고정된 시그너처를 이용하여 바이러스를 탐지 불가능하게 한다. 폴리모픽 웜은 외양을 변경시킨다. 다시 말하면 동일한 기능은 유지하며 웜 코드를 변경시키는 것이다. 메타모픽 웜은 웜의 행위를 동적으로 변경 시킨다. 즉 웜의 기능을 사실상 변경하는 것이다. 폴리모픽과 메타모픽 수행능력을 포함하기위해서 익스플로잇/웜 탄두, 전파엔진, 폴리모픽 엔진으로 변경된 전체 웜, 페이로드를 씌우기 위한 메타모픽 도구, 타깃 선택 알고리즘, 스캐닝 엔진, 암호화된 페이로드 같은 웜 컴포넌트가 포함된다.

폴리모픽 웜은 공격 벡터로 흔히 버퍼 오버플로를 이용하고 있다. 이러한 웜의 폴리모픽 복호기는 보통 짧다. IA(Intel Architecture) 32 구조상에서 기본적인 폴리모픽 복호기는 평균 110바이트이다. 웜 코드를 익스플로잇의 일부로 만드는 웜의 구조는 버퍼를 오버 플로하기에 충분히 많은 양의 데이터를 전송할 필요가 있을 때 적당하다. 이것을 기반으로 폴리모픽 웜 구현의 필요한 부분을 기술한다.

• 공격 벡터

2003년에 보고된 윈도우 미디어 서비스에 대한 기존의 익스플로잇 사용으로, 포트 80을 목표로 하는 단순한 스택 오버플로 익스플로잇이 있다. (그림 2)는 오버플로를 일으키는 HTTP request의 예를 보여 준다.

```
POST scripts/nsiislog dll HTTP/1.1<CR> <LF>
Accept: */* <CR> <LF>
Content-Type : text/plain <CR> <LF>
Content-Length : 9996 <CR> <LF>
<CR> <LF>
...
<long string>
...
<CR> <LF>
```

(그림 2) 오버 플로를 일으키는 schematic 예

• 폴리모픽 복호기

단순한 폴리모픽 복호기(PD: Polymorhic Decryptor)를 사용하며, PD의 목표는 웜의 몸체를 복호화하여 제어를 웜의 로더에게 넘기고, 웜의 로더는 웜을 디스크에 쓰고 실행한다.

• 웜 로더

PD가 완료된 후 제어가 웜 로더로 전환된다. 웜 로

더의 목표는 복호화된 웜/PE의 내용을 임시파일에 저장하고 실행하기 위한 것이다. 이를 위해 4개의 윈도우 API 함수: CreateFile(), WriteFile(), CloseHandle(), CreateProcess()를 사용하였다 [3].

• 단순 웜/PE

비주얼 C++.Net으로 표본 웜/PE 실행코드(executable)를 구현하였다. PE는 아주 단순한 구조이며 CLET 엔진[4]을 기반으로 한다. 하나의 쓰레드로 non-blocking 소켓을 사용하며, 새로운 감염에 대해 웜은 메모리상에 익스플로잇 복사본을 구성하고, 로더를 정적 버퍼로부터 익스플로잇으로 복사한다. PD를 위해 3개의 레지스터, 키와 변환을 포함하는 구조 생성, 그리고 파일내용을 메모리에 복사한 후 변환과 키를 이용하여 자신을 암호화하고, 대응되는 PD를 생성하고 그것을 버퍼 속에 넣고, 마지막으로 임의 IP 주소를 생성하여 그 주소로 익스플로잇 전송을 시도한다.

폴리모픽 복호기를 구문적으로 폴리모픽하게 만들기 위하여 사용되는 기본적인 기법은 아래와 같다 [3].

- 의미 있는 명령어들 Do-nothing 명령어 사이에 끼우기
- 같은 결과를 가져오는 다른 명령어 사용
- PD의 각 버전에 사용되는 레지스터 집합 뒤섞기(shuffling)
- PD가 실행될 때 일부를 복호화하고 재암호화
- 여러 계층의 복호기 사용
- PD 코드의 독립적인 부분을 분리하고 병행 혹은 임의 순서 실행
- 암/복호화에 타이밍 파라미터를 사용하여 디버거 하에서 PD를 수행하여 야기되는 과도한 지연 발생
- 다른 컴퓨터상에 단지 한번만 사용되는 복호화 키 저장
- 시스템 파라미터를 키로 사용

III. 폴리모픽 웜 엔진

3.1 개요

폴리모피즘은 시그너처와 같은 패턴 매칭에 대항하는 방법이다. 폴리모픽 엔진은 대부분 버퍼 오버 플로 취약성을 이용한다. 버퍼 오버플로 코드는 (그림 3)과 같이 4개의 부분으로 이루어져 있다.

NOP	shellcode	bytes to cram	return address

(그림 3) 버퍼 오버플로 코드

취약점을 증명하기 위해 해커들은 해당제품 또는 연구대상에 대한 간단한 설명과 자신이 발견한 취약점이 어떠한 원리에 의해서 발생되는지를 밝히는 것이 통상적이다. 이때 가장 자주 활용되는 방법이 POC(Proof of Concept) 또는 실제로 공격을 수행할 수 있는 코드를 제공한다. 이렇게 제공되는 코드에 공통적으로 들어가는 것으로, 해당 취약점이 발생하도록 구현하는 부분과 이를 이용하여 권한 상승이나 엉뚱한 포트에 백도어를 생성하는 등의 일을 수행하는 것을 셸 코드라 한다.

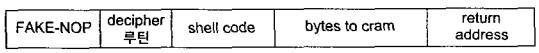
최근 NIDS는 연속적인 NOP를 발견하고 위조된 NOP존을 탐지했다고 믿을 때, 셸 코드상에서 패턴 매칭을 만들기를 시도한다. 이것은 효율적인 방법은 아니지만 버퍼를 채우는 바이트 부분과 또는 수많은 연속적 반환 주소를 인식하기 위한 방법을 생각할 수 있다. 이에 폴리모픽 엔진은 그런 부분들을 인식할 수 없도록, 아래와 같은 방법이 사용된다.

- NOP 계열은 1,2,3 바이트의 무작위 명령의 계열 안에서 변화 된다.
- 셸 코드는 암호화되고 복호화 루틴은 무작위로 생성되어진다.
- 폴리모픽 셸코드 안에서 큰 반환주소 존을 만들지 않는다. 탐지를 무효화하기 위해서 주소부분 크기를 제한한다.
- 반환주소를 숨기는 ADMutate 도구는 반환주소를 불확실성을 가지고 선택한다. ADMmutate는 매번 발생 때 반환주소의 낮은 가중(low-weight) 비트를 변경한다.

셸코드는 바이러스와 같은 것은 아니지만 아래와 같은 속성이 있다.

- 셸코드는 가장 마지막에 실행된다. 따라서 레지스터를 저장할 필요가 없다. 이러한 점을 이용하여 fake-nop(임의의 한 바이트 명령어 코드)에서 아무것도 만들지 않는 코드를 생성시키지 않는다. 예로는 INCR & DECR, ADD & SUB, PUSH & POP 등이 있다.
- 임의의 복호화 방법은 다형태적 이어야한다.
- 셸 코드는 내부에 영(zero)을 가져서는 안 된다. 항상 코드 저장을 위해서 스트링을 사용한다.

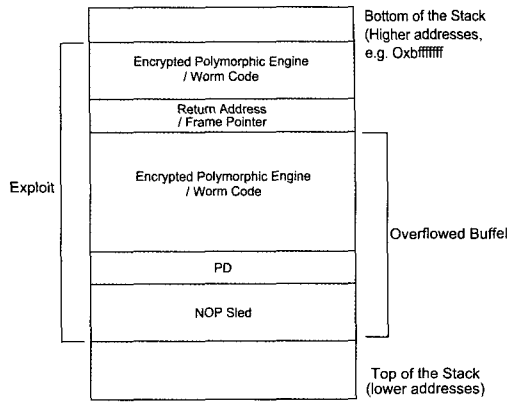
(그림 4)는 폴리모픽 셸 코드의 구성을 보여준다[4].



(그림 4) 폴리모픽 셸 코드 구성

(그림 5)는 폴리모픽 웜을 포함하고 있는 스택 기반 익스플로잇 예를 보여준다.

폴리모픽 웜은 자신의 외관을 변화시켜 바이트 시퀀스가 완전히 달라 보이나, 실제코드는 보통 같다.



(그림 5) 스택 기반 익스플로잇 예

폴리모픽 바이러스와 유사한 방법을 사용할 수 있다. 한 방법은 웜의 원래 코드를 가지고 랜덤키로 암호화하고, 그 키에 대해 짧은 복호기를 생성하는 것이다. 폴리모픽 복호기와 그 키는 매번 바뀐다. 이 작업은 폴리모픽 엔진에 의해 수행된다. 현재 익스플로잇 변화를 위한 도구로는 ADMmutate, CLET, 그리고 JempiScodes 등이 있다[1,4,5].

3.2 ADMmutate

어떠한 서비스에도 버퍼 오버플로 공격이 가능한 도구이다. 폴리모픽 셸 코드의 결과인 익스플로잇 시그너처를 변경하는 것이 목적이다. 버퍼 오버플로는 프로그램이 처리하는 입력에 대해 적당한 검사를 수행하지 않는 프로그래밍 오류이다. 프로그래밍에 오류를 발생시켜 슈퍼 유저 특권으로 실행되기까지 한다면, 공격자는 그 프로세서의 제어를 전복시킬 수 있다. 일단 공격자가 버퍼 오버 플로 오류가 있는 특권이 있는 프로그램을 발견하면 버퍼 오버 플로 공격을 구성하려고 시도한다. 다음은 버퍼 오버플로 익스플로잇의 구성 요소이다.

- 프로세서 NOOP/NOP(no operation) 명령어 :

공통된 NOOP 명령은 0x90의 16진 값을 가진다. 이 명령은 공격자에 의해 선택된 장소로 프로세서의 명령어 포인터(IP)를 전진시키는 NOOP sled 안에서 함께 같이 연결된다.

- 셀코드 명령어 : 공격자에게 원거리 접속을 허용하기 위한 실제적인 어셈블리 명령어이다. 가장 일반적인 셀코드 명령은 (/bin/sh와 같은) 셀을 실행하는 것이다.
- 반환 주소 : 대상 기계의 컴퓨터 메모리에 올바른 주소를 겹쳐 써는 공격자-공급 값이며, 이 값을 정확하게 획득하는 것이 버퍼 오버플로 익스플로잇 전개 첫 단계이다.

ADMmutate 특징은 XOR 암호화를 사용하는 것이며, 가중치 XOR 뿐만 아니라 ADD/SUB, 암호화/복호화를 위해 ROR/ROL을 사용한다. 과다한 NOP 영역, XOR 셀 코드 암호화, 정크 삽입 등이 특징이다. 폴리모픽 셀 코드의 구성은 (그림 6)과 같다.

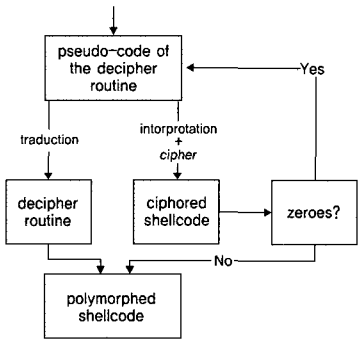
```
[NNNN NNNN NNNN] [SS SS] [RR RR]
N = NOOP 명령어 (응용이 충분한 영역을 준다면 1000 bytes까지 가능)
S = Shellcode 명령어 (보통 80~200 bytes 정도)
R = Return address (보통 200 bytes 정도)
```

(그림 6) 폴리모픽 셀 코드 구성

많은 NOOP 명령어(55개)가 있고 XOR을 사용하여 암호화하고, 그리고 복호화 엔진이 분석을 피할 수 있는 시도로 구축된다. ADMmutate 도구 사용법은 두 가지 방법이 있다. 첫 번째는 폴리모피즘 익스플로잇 코드화를 위해 API를 제공하는 것이고, 두 번째로는 m7 프로그램 익스플로잇 필터를 활용하는 것이다. m7 프로그램은 셀코드를 입력하여 암호화해서 출력한다.

3.3 Clet

Clet 폴리모픽 셀코드 엔진의 원리는 매번 다른 루틴을 사용하여, 임의의 장소에 임의의 키를 가지고 XOR를 생성하고, 그 다음에 원하는 가역적인 명령어(ADD/SUB, ROR/ROL)들을 생성한다. 어셈블리 언어대신 의사코드(pseudo)-어셈블리 언어를 사용하여 생성한다. 다음 그림 7은 Clet의 동작과정을 보여준다.



(그림 7) Clet 동작과정

셀코드의 끝에 보충하는 바이트(cramming byte)를 추가함으로써 바이트의 빈도를 조절한다. 이렇게 함으로 시그너처 기반 시스템이 익스플로잇을 탐지하는 것을 어렵게 한다. 예를 들어, 어떤 시그너처-기반 IDS는 "NOP sleds" 혹은 스택을 오버 플로하기 위하여 사용된 일련의 단일 바이트 NOP 명령어를 찾는다. 이렇게 함으로써 반환 주소의 범위를 추측할 수 있다. 만약 반환 주소를 정확하게 추측하지 못하고 NOP 명령어들의 순서를 히트하면 실행은 공격 코드까지 가게 된다. 다른 IDS 접근으로는 /bin/sh 문자열을 찾는다. 이 변환 도구는 IDS가 통상적인 시퀀스 탐지를 하는 것을 어렵게 만든다.

Clet 엔진은 단일 쓰레드이며 논-블록킹 소켓을 사용한다. 감염에 대해 메모리에 있는 익스플로잇 복

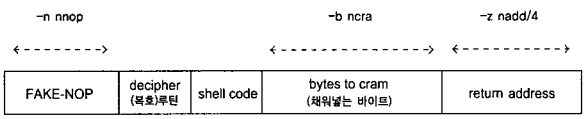
사본을 구성하고, 정적 버퍼에서 익스플로잇 안으로 로더로 복사하며 그리고 폴리모픽 복호화를 위해 임의의 3개의 레지스터, 키와 변형을 포함하는 구조를 생성한다. 다음에 메모리로 파일의 내용을 복사하고 변환과 키를 사용하여 암호화한다. 그런 후 대응 플리모픽 복호기를 생성시켜 버퍼에 삽입한다. 마지막으로 무작위로 IP를 생성하여 익스플로잇 전송을 시도한다. 다음 (그림 8)은 Clet 폴리모픽 구문(syntax)을 보여준다. 버전은 1.0.0이다.

```
syntax :
   /clet

-n  nnop     : nnop NOP 생성
-a           : NOP를 생성하기 위해서 미국영어사전을 사용
-c           : 버퍼의 C 형식 출력
-i  nint     : 복호화 루틴은 9개의 명령어를 가짐(기본값 5)
-f  file     : 폴리로픽에 사용될 스펙트럼 파일
               스펙트럼을 사용하든지 안하든지 ncra 채워 넣는 바이트
-b  ncra     : 생성
-B           : 바이트를 채워 넣는 지역이 시작부로 조정 된다.
-t           : 생성된 바이트 수는 4의 배수
-x  XXXX     : XXXX는 주소 지역에 대한 주소(예:FE011EC9)
-z  nadd     : nadd * 4 바이트의 주소 지역 생성
-e           : shellcode 실행
-d           : stdout으로 shellcode 덤프
-s           : 스펙트럼 분석
-S file      : 파일로부터 shellcode 적재
-E [1-3]     : 임베디드 shellcode 적재
-h           : 메시지 디스플레이
```

(그림 8) Clet 폴리모픽 구문

(그림 9)는 Clet 도구의 바이트 단위의 크기 선택사항들을 보여준다.



(그림 9) Clet 도구의 바이트 단위의 크기 선택사항

3.4 TAPiON

TAPiON은 폴리모픽 복호기 생성기이며, 엔진은 코드의 탐지를 회피하기위해 개발되었다. 유일한 복호 알고리즘을 생성하고 원본 자료를 암호화하고, 코드가 실행되는 동안 on-the-fly로 복호화 가능하다. TAPiON의 특징은 다음과 같다[6].

- 임의로 생성된 복호기에 기반한 복호키
- CPU 시간에 기반한 복호화(임의 선택), 항(anti) - 에물레이터 코드
- RDTSC/coprocessor 명령어 사용 -항 에물레이터 코드
- 키 증가의 임의 단계
- 임의의 레지스터 사용
- 복수 명령 변형(variant)
- 블록 스와핑
- garbage 엔진
- 임의의 복호기 크기
- 복수 복호기 계층 생성

앞에서 기술한 폴리모픽 웜 엔진이외에 JempiScodes, EE1, EE2, EE3 등이 있다. JempiScodes는 기본적으로 exploit 안에서 shellcode의 마지막 기능을 차단할 수 있다는 아이디어에서 나온 것이다[5]. 특징은 NOP 영역 발생이 없고, XOR 암호화 그리고 4개의 다른 암호화 방법이 있는 것이다. EE1은 XOR 암호화와 JUNK 명령어를 사용하고, EE2는 TEA 암호화와 역시 JUNK 명령어를 사용하는 것이다. 마지막으로 EE3는 암호화에 다른 명령의 용법을 사용하며, 이것 역시 JUNK명령어를 사용한다.

IV. 폴리모픽 웜 탐지 기술

폴리모픽 웜의 경우 스트링 지문(string fingerprinting)을 이용한 웜의 탐지에 문제가 있다. Polygraph 솔루션에서는 폴리모픽 웜의 모든 관측에 공

통적인 변하지 않는 다수의 바이트 스트링 포획을 제안하고 있다. 프로토콜 프레이밍 스트링과 버퍼 오버플로에는 일정한 연속적인 바이트 스트링들이 웜 실행 시에 항상 일정하게 유지 되어 웜 시그너처 생성에 사용될 수 있음을 보여준다[7].

폴리모픽 웜을 탐지하기위해서 통계적 측정을 기반으로 하는 PADS(Position-aware distribution signatures) 시그너처는 지수 1<n<=m을 가진 순서로 된 바이트-빈도 분포 시퀀스이다[8]. 각 바이트 빈도 분포는 악성 트래픽에 포함된 바이트의 시퀀스에 존재할 바이트 값의 가능성을 정의한다. 여기서 핵심은 폴리모픽 익스플로잇의 모든 바이트 값이 변하더라도 바이트 값의 통계 분포는 모든 변화에 대해 유사하다는 것이다.

Earlybird 시스템은 웜의 일반적인 행위에 의존하며 웜에 생성된 하나의 네트워크 패킷은 변하지 않는 바이트-시퀀스를 포함한다는 가정에 기반을 두고 있다[9].

4.1 지문 생성(Finger-Printing) 기술

가. 실행코드 구조정보 이용방법

이 방법은 웜이 적어도 어떤 부분이 실행가능한 머신 코드를 포함하고 있다고 전제 한다. 실행코드와 관심 영역이 네트워크 내부에서 식별될 때, 이 영역에 대하여 핑거프린트(지문)를 생성한다. 핑거 프린팅은 콘텐츠 시프팅(Content Sifting) 방법에 의해 네트워크 스트림으로부터 추출되는 바이트 스트링과 관련이 있다. 악성코드의 단순한 수정으로 피할 수 없는 상위레벨의 추상화에서 지문을 생성한다. 지문 생성기법의 요구 사항은 다음과 같다:

- 유일성(Uniqueness) : 다른 실행코드 영역은 다른 지문으로 사상되어야 한다는 성질이다. 이 속성이 만족되지 않으면, 시스템은 오탐(false positives)을 일으키기 쉽다.
- 삽입(insertion)과 삭제(deletion)에 대한 견고성 : 공격자가 실제적인 악성 코드 단편 전이나 후에 코드를 삽입하여 회피 시도에 대한 대응을 하기 위하여 필요하다.
- 변조에 대한 견고성 : 어떤 연산에 의하여 코드 시퀀스가 변조될 때에도 결과적인 지문이 같아야 되는 성질이다. 이 속성은 단일 폴리모픽 웜의 다른 변화를 식별하기 위하여 필요하다.

실행코드의 구조는 제어흐름그래프(CFG: Control Flow Graph)에 의해 기술되며, 그래프는 정적인 플로분석으로 추출된 머신 코드에서 생성되며 분기와 점프명령에 의해 정의된다. 시그너처는 추출된 CFG를 기반으로 하며 모든 시그너처는 특정 그래프와 일치한다.

나. 자동 웜 지문생성 기술

이 기법은 새로운 웜을 탐지하는 것이고 탐지된 웜에 대해 자동적으로 시그너처를 생성하는 것으로 Earlybird로 불린다. 전형적인 웜의 행위와 그 웜에 의해 생성된 하나의 네트워크 패킷은 변하지 않는 바이트-시퀀스를 포함한다는 가정을 기반으로 한다. 이 기법은 아래와 같은 가정에 의존한다.

- 콘텐츠 불변(content invariance) : 웜에 의해 생성되는 네트워크 패킷은 변하지 않는 바이크-시퀀스를 포함한다.
- 콘텐츠 유포(content prevalence) : 웜의 자기 복제 행위 때문에 변하지 않는 바이트-시퀀스는 다른 포획 네트워크 패킷들에서 빈번히 발생한다.
- 주소 확산(Address dispersion) : 시간이 경과할수록 감염 호스트가 증가한다. 결과적으로 불변 바이트 시퀀스를 포함하는 패킷을 근원지로

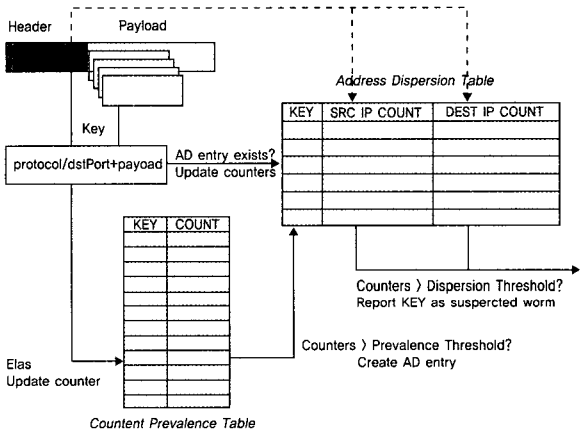
하는 IP 주소가 증가한다.

이 방법은 웜을 탐지하고 시그너처를 생성하기위해 네트워크-단계 데이터를 사용한다. 네트워크 내에서 포획된 모든 패킷들은 단일 센서에 의해 분석되며, 센서는 변하지 않는 페이로드를 포함하고 있는 패킷 혹은 바이트 시퀀스 탐지를 시도한다. 계산적인 복잡성 감소를 위해서 작고 고정된 길이의 바이트-시퀀스에 초점을 두고 있으며, 기술적으로 모든 고정된 길이의 바이트-시퀀스의 발생은 자주 일어나는 시퀀스를 탐지하기위하여 유포 테이블에 저장된다. 동일성을 식별하기위해서 각 시퀀스를 다항식에 할당하는 라빈(Rabin) 지문의 변화를 사용할 것을 제안한다. 따라서 두개의 같은 시퀀스는 같은 지문을 생성한다. 빈번하게 발생하는 바이트-시퀀스가 식별되면 웜 시그너처로 사용한다.

네트워크 웜은 확산으로 많은 트래픽을 생성하고, 이 트래픽은 공통 서브스트링을 포함한다. 트래픽은 다양한 다른 근원지와 목적지로 향하며, 트래픽 패턴 식별은 웜 탐지를 위해서 충분하다고 가정한다. 각 네트워크 패킷에 대해 콘텐츠는 추출되고 모든 서브스트링은 처리된다. 각 서브스트링은 매번 발견될 때마다 카운트 항목을 증가시켜 유포 테이블에 색인된다. 테이블에 관측된 서브스트링은 모두 히스토그램으로 구현되며 유일한 근원지와 목적지 주소의 카운트의 유지를 위해 두개의 목록을 유지한다. 이 목록은 검색되고 서브스트링 카운트가 증가될 때 마다 갱신된 주소이다. 서브스트링 카운트와 주소 목록의 크기에 대해 이 목록을 분류하면 웜 트래픽 일 것 같은 집합이 생성된다. 웜 행위의 기준에 부합되는 테이블 항목들은 웜의 불변 서브스트링을 포함하는 것으로 이 서브스트링들이 합법적인 네트워크 트래픽에서 웜을 걸러내는 시그너처로 사용된다. 이러한 기술을 콘텐츠 시프팅이라 하며, 네트워크 트래픽의 콘텐츠 상에 고역 여파기를 효과적으로 구현한다. 유포되지 않거나 넓게 확산되지 않는 네트워크 콘텐츠는 가려내고, 웜 같은 컨텐츠만 남는다.

위는 이상적인 콘텐츠 시프팅 기술이며, 실제적인 콘텐츠 시프팅 기술로 콘텐츠 유포 평가 기술, 주소 확산 평가 기술, CPU 스켈링(scaling) 등에 대하여 기술하고 있다.

(그림 10)은 EarlyBird 원형에서 구현된 콘텐츠 시프팅 알고리즘을 보여준다.



(그림 10) EarlyBird에서 사용된 콘텐츠 시프팅 알고리즘

각 패킷이 도착할 때, 콘텐츠 해시 코드를 생성하기위해서 그것의 콘텐츠가 해시되고 프로토콜 식별자와 목적지 주소가 추가된다. 구현에서 패킷 해시로 32비트 CRC를, 서브스트링 해시를 위해서 40바이트 라빈(Rabin) 지문을 사용하고 각 라빈 지문은 f=1/64로 서브 샘플된다. 결과의 해시코드는 주소확산테이블을 인덱스하기 위해서 사용된다. 만약 엔트리가 이미 존재한다면 근원지와 목적지 IP 주소에 대한 주소 확산 테이블 엔트리는 갱신된다. 만약 근원지와 목적지 카운트가 확산 임계치를 초과하면, 콘텐츠 스트링이 보고된다.

콘텐츠 해시가 확산 테이블에서 발견되지 않으면 콘텐츠 유포 테이블 쪽으로 인덱스 된다. 4개의 카운터 배열 쪽으로 4개의 색인을 생성하여 콘텐츠 해시의 4개의 독립적인 해시함수를 사용한다. 보존적 갱신 최적화를 사용하여 4개의 카운터들 사이 가장 적은 것이 증가된다. 만약 4개의 카운터가 유포 임계치보다 더 크다면 새로운 엔트리는 주소 확산테이블에서 만들어진다. (그림 11)은 EarlyBird 시스템의 주요 반복하는 의사코드이다.

```
ProcessPacket()
1   InitializeIncrementalHash(payload,payloadLength,dstPort)
2   while (currentHash=GetNextHash())
3     if (currentADEntry=ADEntryMap.Find(currentHash))
4       UpdateADEntry(currentADEntry,srcIP,dstIP,packetTime)
5       if ( (currentADEntry.srcCount > SrcDispTh)
              and (currentADEntry.dstCount > DstDispTh) )
6         ReportAnomalousADEntry(currentADEntry.packet)
7       endif
8     else
9       if ( MsfIncrement(currentHash) > PrevalenceTh)
10        newADEntry=InitializeADEntry(srcIP,dstIP,packetTime)
11        ADEntryMap.Insert(currentHash,newADEntry)
12      endif
13    endif
14  endwhile
```

(그림 11) EarlyBird 반복 의사코드

4.2 의미론-인지(Semantics-Aware) 탐지 기술

오용 탐지를 위한 네트워크 기반 침입탐지시스템은 이전 문서화된 공격 프로파일을 식별하기 위해 구축된 시그너처 데이터베이스 혹은 패턴을 이용하여 트래픽을 비교한다. 오용탐지의 효율성은 시그너처 데이터베이스의 품질과 직접적으로 연결되어 시그너처 관리와 생성을 어렵게 만든다. 시그너처는 특정적(specific)이어야하고, 특정한 공격 프로파일의 특성을 식별하여야 한다. 오탐을 일으키는 특정성(specificity)의 결여가 NIDS의 주요한 문제 중의 하나이다.

상업적인 멀웨어 탐지는 단순한 패턴 매칭을 사용한다. 프로그램이 만약 규칙적인 표현식의 명령어 순서를 포함한다면 멀웨어로 선언 된다. 멀웨어를 탐지하는 패턴 매칭의 기본적 결여는 명령어의 의미론(semantics)을 무시하는 것이다. 패턴 매칭 알고리즘은 가벼운 변화에 탄력적이지 않으므로 멀웨어 탐지는 각기 다르고 가볍게 변화하는 두개의 멀웨어 인스턴스 탐지를 위해서 다른 패턴을 사용해야 한다. 이런 이유로 상업적인 바이러스 스캐너의 시그너처 데이터베이스는 때때로 갱신된다.

확실한 멀웨어의 모든 변화가 나타나는 의심스러운 행위를 확실하게 관찰 하는 것에 형식적 의미론(Formal Semantics)과 의미론-인지 멀웨어 탐지 알고리즘(Semantics-Aware Detection Algorithm)을 사용한다[10].

- 형식적 의미론: 프로그램이 명세된 의심스러운 행위를 표시하는지를 결정하는 문제를 형식적으로 정의한다.
- 의미론-인지 멀웨어 탐지 알고리즘: 프로그램에 명세된 의심스러운 행위를 나타내는지가 분명한지 어떤지를 결정하기위해 사용한다.

V. 결론 및 향후 계획

악성 웜은 확산 능력과 손상 규모를 증가시키면서 지속적인 진화를 하고 있다. 앞으로의 웜은 다양한 파괴적인 특성을 가질 것이며, 멀티 플랫폼, 멀티 익스플로잇, zero-day, 폴리모픽, 메타모픽 등의 형태가 될 것으로 추측된다. 폴리포픽 웜은 확산되는 동안 탐지를 피할 수 있도록 웜의 외양이 변하는 능력을 가진 웜이다. 이에 대한 폴리모픽 엔진 도구들은 AD Mutate, Clet, Jempiscode, Tapion 등이 있으며, 이들 엔진을 이용하면 폴리모픽 웜 제작이 용이하다.

본 논문에서는 비정상 트래픽에 의한 새로운 네트

워크 공격 유형에서 폴리모픽 웜에 대해 알아보았다. 폴리모픽 웜에 대한 기초적인 분석 및 조사 결과를 요약하여 기술 하였고, 폴리모픽 웜의 정의와 구조, 웜의 구현 방법에 대하여 기술하였다. 또한 폴리모픽 웜 생성 원리 및 주요한 폴리모픽 엔진에 대하여 기술하였다. 폴리모픽 웜의 탐지 기술로서 지문 생성 기술과 의미론-인지 기술에 대하여 논의하였다. 향후 연구에서는 폴리모픽 웜 엔진의 동작원리와 탐지 기술과 대응에 대하여 좀 더 체계적인 분석을 수행 할 예정이다[11-15].

[참 고 문 헌]

[1] K2, ADMmutate, http://www.ktwo.ca/security.html.

[2] David J. Albanese, Michael J. Wiacek, Christopher M. Salter, and Jeffrey A. Six, The Case for Using Layered Defenses to Stop Worms, Report #C43-002R-2004, Version 1.0, June 18, 2004, National Security Agency.

[3] O. Kolesnikov, D. Dagon, and W. Lee, "Advanced Polymorphic Worms : Evading IDS by blending in with normal traffic", College of Computing, Georgia Inst. of Tech, Atlanta, GA, 2004.

[4] CLET Team, "Polymorphic Shellcode Engine Using Spectrum Analysis", http://www.phrack.org, Phrack 61/9, 2003.

[5] M. Sedalo, Jempiscodes: Polymorphic shellcode generator, 2003. http://securitylab.ru/tools/services/download/?ID=36712.

[6] Tapion Project, http://pb.specialised.info/all/tapion/

[7] J. Newsome, B. Karp, and D. Song. Polygraph: Automatic signature generation for polymorphic worms. In Proceedings of the 2005 IEEE Symposium on Security and Privacy, May 2005.

[8] Y. Tang and S. Chen. Defending against Internet worms: A signature-based approach. In Proceedings of the IEEE Infocom 2005, Miami, Florida, USA, May 2005.

[9] S. Singh, C. Estan, G. Varghese, and S. Savage. Automated worm fingerprinting. In Proceedings of the ACM/USENIX Symposium on Operating System Design and Implementation, San Francisco, SA, USA, December 2004.

[10] Mihai Christodorescu et al., "Semantics-Aware Malware Detection". (U of Wisconsin & CMU).

[11] C.Kruegel, E.Kirda, D.Mutz, W.Robertson, and G. Vigna. Polymorphic worm detection using structural information of executables. In Proceedings of the 8th International Symposium on Recent Advances in Intrusion Detection(RAID), September 2005.

[12] A. Pasupulati et al., "Buttercup: On network-based detection of polymorphic buffer overflow vulnerabilities", In 9th IEEE/IFIP Network Operation and Management Symposium (NOMS'2004).

[13] U. Payer, P. Teufl, and M. Lamberger, "Hybrid engine for polymorphic shellcode detection", In Proc. of DIMVA, 2005.

[14] Ed Skoudis and Lenny Zeltser, Malware: Fighting Malicious Code, Prentice-Hall, 2004, (Chapter 2: Virus, 3: Worm)

[15] V. Yegneswaran, J. T. Giffin, P. Barford, and S. Jha, "An architecture for generating semantics-aware signature", In USENIX Security Symposium, 2005.

전용희

1971년 ~ 1978년 고려대학교 전기전자전파공학부
1985년 ~ 1987년 미국 플로리다공대 대학원 컴퓨터공학과
1987년 ~ 1992년 미국 노스캐롤라이나주립대 대학원 Elec. and Comp. Eng. 석사, 박사
1978년 ~ 1978년 삼성중공업(주)
1978년 ~ 1985년 한국전력기술(주)
1979년 ~ 1980년 벨기에 벨가톰(Belgatom)
1989년 ~ 1989년 미국 노스캐롤라이나주립대 Dept of Elec. and Comp. Eng. TA
1989년 ~ 1992년 미국 노스캐롤라이나주립대 부설 CCSP(Center For Comm. & Signal Processing) RA
1992년 ~ 1994년 한국전자통신연구원 광대역통신망연구부 선임연구원
1994년 ~ 현재 대구가톨릭대학교 컴퓨터 · 정보통신공학부 교수
2001년 ~ 2003년 동 공과대학장 역임
2004년 ~ 2005년 한국전자통신연구원 정보보호연구단 초빙연구원
관심분야 : 네트워크 보안, 웜 탐지 및 대응 기술, BcN 보안 및 QoS 보장 기술

장정숙

1989년 ~ 1991년 경일대학교 공과대학 컴퓨터공학과 (공학사)
1992년 ~ 1995년 대구가톨릭대학교 교육대학원 전자계산교육전공(석사)
1998년 ~ 2004년 대구가톨릭대학교 대학원 컴퓨터 · 정보통신공학 전공 (이학박사)
2004년 ~ 현재 대구가톨릭대학교 컴퓨터정보통신공학부 IT교수
관심분야 : 임베디드 네트워크 보안, BcN & QoS 보안, 홈네트워크 보안, 통신망 성능분석

장종수

1984년 경북대학교 전자공학과 공학사
1986년 경북대학교 전자공학과 공학석사
2000년 충북대학교 컴퓨터공학과 공학박사
1989년 ~ 현재 한국전자통신연구원 정보보호연구단 네트워크보안그룹 그룹장
관심분야 : 네트워크보안, 웹서비스보안, Secure OS, IDS/IPS, Traffic Management

남택용

1987년 충남대학교 계산통계학과 이학사
1990년 충남대학교 계산통계학과 이학석사
2005년 한국외국어대학교 전자정보공학과 공학박사
1987년 ~ 현재 한국전자통신연구원 정보보호연구단, 보안게이트웨이연구팀 팀장(책임연구원)
관심분야 : 개인정보보호, 콘텐츠보안, 정보분류 등